Red Hat OpenStack Platform 16.1

イメージの作成と管理

イメージの作成と管理

Last Updated: 2023-09-19

Red Hat OpenStack Platform 16.1 イメージの作成と管理

イメージの作成と管理

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドでは、イメージを作成および管理する手順、ならびに Image サービスを設定する手順について説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. オプション: ドキュメントチームが問題の詳細を確認する際に使用できるメールアドレスを記入してください。
7. **Submit** をクリックします。

第1章 IMAGE サービス

Red Hat OpenStack Platform (RHOSP) でのイメージおよびストレージを管理します。

仮想マシンのイメージとは、ブート可能なオペレーティングシステムがインストールされた仮想ディスクが含まれるファイルです。仮想マシンのイメージは、複数の形式をサポートしています。以下は、RHOSP で利用可能な形式です。

- **RAW**: 非構造化のディスクイメージ形式
- **QCOW2**: QEMU エミュレーターでサポートされているディスク形式。この形式には、QEMU 1.1 以降が必要な QCOW2v3 (QCOW3 と呼ばれる場合があります) が含まれます。
- **ISO**: ディスク上のデータをセクター単位でコピーし、バイナリーファイルに格納した形式
- **AKI**: Amazon Kernel Image
- **AMI**: Amazon Machine Image
- **ARI**: Amazon RAMDisk Image
- **VDI**: VirtualBox の仮想マシンモニターおよび QEMU エミュレーターでサポートされているディスク形式
- **VHD**: VMware、VirtualBox などの仮想マシンモニターで使用されている一般的なディスク形式
- **PLOOP**: OS コンテナーを実行するのに Virtuozzo でサポートおよび使用されているディスク形式
- **OVA**: Image サービス (glance) に保存されているのが OVA tar アーカイブファイルであることを示します。
- **DOCKER**: Image サービス (glance) に保存されているのがコンテナーファイルシステムの Docker tar アーカイブであることを示します。

通常、仮想マシンイメージの形式に **ISO** は考慮されませんが、ISO にはオペレーティングシステムがインストール済みのブート可能なファイルシステムが含まれているので、他の形式の仮想マシンイメージファイルと同様に使用されます。

公式の Red Hat Enterprise Linux クラウドイメージをダウンロードするには、お使いのアカウントに有効な Red Hat Enterprise Linux サブスクリプションが必要です。

- Red Hat Enterprise Linux 8 KVM Guest Image
- Red Hat Enterprise Linux 7 KVM Guest Image
- Red Hat Enterprise Linux 6 KVM Guest Image

カスタマーポータルにログインしていない場合には、プロンプトが表示されるので Red Hat アカウントの認証情報を入力する必要があります。

1.1. IMAGE サービスについての理解

Red Hat OpenStack Platform (RHOSP) Image サービス (glance) の機能

1.1.1. サポート対象の Image サービスバックエンド

以下に示す Image サービス (glance) バックエンドのシナリオがサポートされます。

- Ceph を使用する場合には、RBD がデフォルトのバックエンドです。詳しい情報は、**オーバークラウドの高度なカスタマイズ**の Ceph ストレージの設定 を参照してください。
- RBD マルチストア。詳細は、ストレージエッジアーキテクチャーの要件 を参照してください。
- Object Storage (swift)。詳しい情報は、**オーバークラウドの高度なカスタマイズ**の 外部の Object Storage クラスターの使用 を参照してください。
  Image サービスは、Object Storage のタイプとバックエンドをデフォルトとして使用します。
- Block Storage (cinder)。詳しい情報は、**オーバークラウドの高度なカスタマイズ**の Image サービス用 cinder バックエンドの設定 を参照してください。
- NFS。詳しい情報は、**オーバークラウドの高度なカスタマイズ**の NFS ストレージの設定 を参照してください。

  重要

  NFS はサポート対象の Image サービス用デプロイメントオプションですが、より堅牢なオプションを利用することができます。
  NFS は Image サービスネイティブではありません。NFS 共有を Image サービスにマウントした場合、Image サービスは操作を管理しません。Image サービスはファイルシステムにデータを書き込みますが、バックエンドが NFS 共有であることを認識しません。

  この種別のデプロイメントでは、ファイル共有に異常が発生しても、Image サービスは要求をリトライすることができません。つまり、バックエンドで障害が発生した場合、ストアは読み取り専用モードに移行するか、ローカルファイルシステムにデータの書き込みを続けます。この場合、データを損失する可能性があります。この状況から回復するには、ファイル共有がマウントされ同期されている状態にし、続いて Image サービスを再起動する必要があります。このような理由により、Red Hat では、Image サービスのバックエンドとして NFS を推奨しません。

  ただし、Image サービスのバックエンドに NFS を使用することを選択した場合には、以下のベストプラクティスがリスクを軽減するのに役立ちます。

  - 信頼性の高い実稼働環境グレードの NFS バックエンドを使用する。
  - コントローラーノードと NFS バックエンドの間に強力で信頼性の高い接続があることを確認してください。レイヤー 2 (L2) ネットワーク接続が推奨されます。
  - マウントされたファイル共有のモニタリングおよびアラート機能を追加する。
  - 基になるファイルシステムのアクセス許可を設定します。書き込み権限は、ストアとして使用する共有ファイルシステムに設定する必要があります。
  - glance-api プロセスが実行されるユーザーおよびグループが、ローカルファイルシステムのマウントポイントに対する書き込み権限を持たないようにしてください。これにより、プロセスはマウントの異常を検出して、書き込みを試みる際にストアを読み取り専用モードに移行することができます。

1.1.2. イメージの署名および検証

イメージの署名および検証により、デプロイ担当者がイメージに署名して、その署名と公開鍵の証明書をイメージの属性として保存できるようにすることで、イメージの整合性と信頼性を保護します。

この機能を利用すると、次のタスクを実行できます。

- 秘密鍵を使用してイメージに署名し、そのイメージ、署名、公開鍵の証明書 (検証メタデータ) への参照をアップロードすることができます。Image サービスは、署名が有効かどうかを検証します。
- Compute サービスでイメージを作成し、Compute サービスがそのイメージに署名し、イメージや検証メタデータをアップロードすることができます。Image サービスは、この場合も、署名が有効であるかどうかを検証します。
- Compute サービスで署名済みのイメージを要求することができます。Image サービスは、イメージと検証メタデータを提供します。これにより、Compute サービスはイメージを起動する前に検証することができます。

イメージの署名と検証の詳細については、**OpenStack キーマネージャーを使用したシークレットの管理** の Glance イメージの検証 を参照してください。

1.1.3. イメージの変換

イメージの変換は、イメージのインポート中にタスク API を呼び出して、イメージを変換します。

インポートのワークフローの一環として、プラグインがイメージの変換機能を提供します。このプラグインは、デプロイ担当者の設定に基づいて、アクティブ化/非アクティブ化することができます。そのため、デプロイ担当者は、デプロイメントに希望のイメージ形式を指定する必要があります。

内部では、Image サービスが特定の形式でイメージのビットを受信します。これらのビットは、一時的な場所に保管されます。次にプラグインが起動されて、イメージを対象のフォーマットに変換し、最終的な保管場所に移動します。タスクが終了すると、一時的な場所は削除されます。このため、Image サービスでは最初にアップロードした形式は保持されません。

イメージの変換についての詳細は、イメージ変換の有効化 を参照してください。

注記

イメージの変換は、イメージをインポートする時にだけトリガーされます。イメージのアップロード時には実行されません。以下に例を示します。

```
$ glance image-create-via-import \
    --disk-format qcow2 \
    --container-format bare \
    --name NAME \
    --visibility public \
    --import-method web-download \
    --uri http://server/image.qcow2
```

1.1.4. 相互運用可能なイメージのインポート

相互運用可能なイメージのインポートワークフローにより、以下の 2 とおりの方法でイメージをインポートすることができます。

- **web-download** (デフォルト) メソッドを使用して、URI からイメージをインポートする。
- **glance-direct** メソッドを使用して、ローカルファイルシステムからイメージをインポートする。

1.1.5. Image サービスのキャッシュ機能を使用したスケーラビリティーの向上

glance-api キャッシュメカニズムを使用して、Image サービス (glance) API サーバーにイメージのコピーを保存し、それらを自動的に取得してスケーラビリティーを向上させます。Image サービスのキャッシュ機能を使用することで、複数のホスト上で glance-api を実行することができます。つまり、同じイメージをバックエンドストレージから何度も取得する必要はありません。Image サービスのキャッシュ機能は、Image サービスの動作には一切影響を与えません。

Red Hat OpenStack Platform director (tripleo) heat テンプレートを使用して、Image サービスのキャッシュ機能を設定します。

手順

1. 環境ファイルの **GlanceCacheEnabled** パラメーターの値を **true** に設定します。これにより、**glance-api.conf** Heat テンプレートの **flavor** の値が自動的に **keystone+cachemanagement** に設定されます。

```
parameter_defaults:
  GlanceCacheEnabled: true
```

2. オーバークラウドを再デプロイする際に、**openstack overcloud deploy** コマンドにその環境ファイルを追加します。

3. オプション: オーバークラウドを再デプロイする際に、**glance_cache_pruner** を異なる頻度に調節します。5 分間の頻度の例を以下に示します。

```
parameter_defaults:
  ControllerExtraConfig:
    glance::cache::pruner::minute: '*/5'
```

ファイルシステムを使い果たす状況を回避するために、ご自分のニーズに合わせて頻度を調節します。異なる頻度を選択する場合は、以下の要素を考慮に入れます。

- 実際の環境でキャッシュするファイルのサイズ
- 利用可能なファイルシステムの容量
- 環境がイメージをキャッシュする頻度

1.1.6. イメージの事前キャッシュ

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能についての詳しい情報は、対象範囲の詳細 を参照してください。

1.1.6.1. 定期的にイメージを事前キャッシュする際のデフォルト間隔の設定

Red Hat OpenStack Platform (RHOSP) director は、**glance-api** サービスの一部としてイメージを事前キャッシュすることができます。デフォルトの間隔は 300 秒です。実際の要件に応じて、デフォルトの間隔を増減することができます。

手順

1. アンダークラウドの環境ファイルの **ExtraConfig** パラメーターを使用して、実際の要件に応じて新しい間隔を追加します。

```
parameter_defaults:
  ControllerExtraConfig:
    glance::config::glance_api_config:
      DEFAULT/cache_prefetcher_interval:
        value: '<300>'
```

<300> を、イメージを事前キャッシュする間隔の秒数に置き換えてください。

2. **/home/stack/templates/** の環境ファイルで間隔を修正したら、**stack** ユーザーとしてログインして設定をデプロイします。

```
$ openstack overcloud deploy --templates \
-e /home/stack/templates/<ENV_FILE>.yaml
```

<ENV_FILE> は、追加した **ExtraConfig** 設定が含まれる環境ファイルの名前に置き換えてください。

重要

オーバークラウドの作成時に追加の環境ファイルを渡した場合には、予定外の変更がオーバークラウドに加えられないように、ここで **-e** オプションを使用して環境ファイルを再度渡します。

openstack overcloud deploy コマンドについての詳しい情報は、**Director のインストールと使用方法** の デプロイメントコマンド を参照してください。

1.1.6.2. 定期的なジョブを使用したイメージの事前キャッシュ

前提条件

定期的なジョブを使用してイメージを事前キャッシュするには、**glance_api** サービスを実行しているノードに直接接続された **glance-cache-manage** コマンドを使用する必要があります。サービスの要求に応答するノードを非表示にするプロキシーは使用しないでください。アンダークラウドは **glance_api** サービスを実行しているネットワークにアクセスできない可能性があるため、最初のオーバークラウドノード (デフォルトでは **controller-0** という名前です) でコマンドを実行します。

前提条件として以下の手順を実施して、正しいホストからコマンドが実行され、必要な認証情報が設定されるようにします。また、**glance-api** コンテナー内から **glance-cache-manage** コマンドが実行されるようにします。

手順

1. アンダークラウドに stack ユーザーとしてログインし、**controller-0** のプロビジョニング IP アドレスを特定します。

```
(undercloud) [stack@site-undercloud-0 ~]$ openstack server list -f value -c Name -c
Networks | grep controller
overcloud-controller-1 ctlplane=192.168.24.40
overcloud-controller-2 ctlplane=192.168.24.13
overcloud-controller-0 ctlplane=192.168.24.71
(undercloud) [stack@site-undercloud-0 ~]$
```

2. オーバークラウドに対して認証するには、**/home/stack/overcloudrc** (デフォルト) に保存されている認証情報を **controller-0** にコピーします。

```
$ scp ~/overcloudrc heat-admin@192.168.24.71:/home/heat-admin/
```

3. **controller-0** に接続します。

```
$ ssh heat-admin@192.168.24.71
```

4. **controller-0** において、**heat-admin** ユーザーとして **glance_api service** の IP アドレスを特定します。以下の例では、IP アドレスは **172.25.1.105** です。

```
(overcloud) [root@controller-0 ~]# grep -A 10 '^listen glance_api' /var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg
listen glance_api
 server central-controller0-0.internalapi.redhat.local 172.25.1.105:9292 check fall 5 inter 2000 rise 2
```

5. **glance-cache-manage** コマンドは **glance_api** コンテナーでしか利用できないため、そのコンテナーに対して実行するスクリプトを作成します。このコンテナーには、オーバークラウドに対して認証するための環境変数がすでに設定されています。**controller-0** の **/home/heat-admin** に、以下の内容でスクリプト **glance_pod.sh** を作成します。

```
sudo podman exec -ti \
 -e NOVA_VERSION=$NOVA_VERSION \
 -e COMPUTE_API_VERSION=$COMPUTE_API_VERSION \
 -e OS_USERNAME=$OS_USERNAME \
 -e OS_PROJECT_NAME=$OS_PROJECT_NAME \
 -e OS_USER_DOMAIN_NAME=$OS_USER_DOMAIN_NAME \
 -e OS_PROJECT_DOMAIN_NAME=$OS_PROJECT_DOMAIN_NAME \
 -e OS_NO_CACHE=$OS_NO_CACHE \
 -e OS_CLOUDNAME=$OS_CLOUDNAME \
 -e no_proxy=$no_proxy \
 -e OS_AUTH_TYPE=$OS_AUTH_TYPE \
 -e OS_PASSWORD=$OS_PASSWORD \
 -e OS_AUTH_URL=$OS_AUTH_URL \
 -e OS_IDENTITY_API_VERSION=$OS_IDENTITY_API_VERSION \
 -e OS_COMPUTE_API_VERSION=$OS_COMPUTE_API_VERSION \
 -e OS_IMAGE_API_VERSION=$OS_IMAGE_API_VERSION \
 -e OS_VOLUME_API_VERSION=$OS_VOLUME_API_VERSION \
 -e OS_REGION_NAME=$OS_REGION_NAME \
glance_api /bin/bash
```

6. source コマンドで **overcloudrc** ファイルを読み込み、**glance_pod.sh** スクリプトを実行して、オーバークラウドのコントローラーノードに対して認証するのに必要な環境変数が設定されている **glance_api** コンテナーに対して実行します。

```
[heat-admin@controller-0 ~]$ source overcloudrc
(overcloudrc) [heat-admin@central-controller-0 ~]$ bash glance_pod.sh
()[glance@controller-0 /]$
```

7. **glance image-list** 等のコマンドを使用して、コンテナーでオーバークラウドに対して認証されたコマンドを実行できることを確認します。

```
()[glance@controller-0 /]$ glance image-list
+--------------------------------------+---------------------------------+
| ID                                   | Name                            |
+--------------------------------------+---------------------------------+
| ad2f8daf-56f3-4e10-b5dc-d28d3a81f659 | cirros-0.4.0-x86_64-disk.img    |
+--------------------------------------+---------------------------------+
()[glance@controller-0 /]$
```

手順

1. 管理ユーザーとして、キャッシュするイメージをキューに追加します。

```
$ glance-cache-manage --host=<host_ip> queue-image <image_id>
```

- <host_ip> を **glance-api** コンテナーが実行されているコントローラーノードの IP アドレスに置き換えます。
- <image_id> をキューに追加するイメージの ID に置き換えます。
  事前にキャッシュするイメージをキューに追加すると、**cache_images** 定期ジョブはキューに追加されたすべてのイメージを同時に事前取得します。

注記

イメージキャッシュは各ノードにローカルなので、Red Hat OpenStack Platform が HA 設定でデプロイされている場合 (3、5、または 7 台のコントローラー)、**glance-cache-manage** コマンドを実行する際に **--host** オプションでホストのアドレスを指定する必要があります。

2. 以下のコマンドを実行して、イメージキャッシュ内のイメージを表示します。

```
$ glance-cache-manage --host=<host_ip> list-cached
```

<host_ip> を環境内のホストの IP アドレスに置き換えてください。

関連情報

以下の目的で、さらに **glance-cache-manage** コマンドを使用することができます。

- **list-cached**: 現在キャッシュされているすべてのイメージをリスト表示する。
- **list-queued**: キャッシュするために現在キューに追加されているすべてのイメージをリスト表示する。
- **queue-image**: キャッシュするためにイメージをキューに追加する。
- **delete-cached-image**: キャッシュからイメージを削除する。
- **delete-all-cached-images**: キャッシュからすべてのイメージを削除する。
- **delete-cached-image**: キャッシュのキューからイメージを削除する。
- **delete-all-queued-images**: キャッシュのキューからすべてのイメージを削除する。

1.1.7. metadef API の保護

Red Hat OpenStack Platform (RHOSP) では、ユーザーはメタデータ定義 (metadef) API を使用してキー/値のペアおよびタグメタデータを定義することができます。現時点では、ユーザーが作成することのできる metadef 名前空間、オブジェクト、属性、リソース、またはタグの数に制限はありません。

metadef API により、情報が権限のないユーザーに漏えいする可能性があります。悪意のあるユーザーは制約がないことを悪用し、Image サービス (glance) のデータベースを無制限のリソースで埋め尽くすことができます。これにより、サービス拒否 (DoS) 型の攻撃を行うことができます。

Image サービスのポリシーは metadef API を制御します。ただし、metadef API のデフォルトのポリシー設定では、すべてのユーザーが metadef 情報を作成または読み取ることができます。metadef リソースへのアクセスは所有者だけに制限されている訳ではないため、内部インフラストラクチャーの詳細や顧客名などの秘匿すべき名前を持つ metadef リソースの情報が、悪意のあるユーザーに漏えいする可能性があります。

1.1.7.1. metadef API を制限するためのポリシーの設定

Image サービス (glance) をよりセキュアにするには、Red Hat OpenStack Platform (RHOSP) デプロイメントのデフォルトでは metadef 変更 API へのアクセスを管理者だけに制限します。

手順

1. クラウド管理者として新たな heat テンプレートの環境ファイルを作成し (例: **lock-down-glance-metadef-api.yaml**)、Image サービス metadef API のポリシーオーバーライドを含めます。

```
...
parameter_defaults:
  GlanceApiPolicies: {
 glance-metadef_default: { key: 'metadef_default', value: '' },
 glance-metadef_admin: { key: 'metadef_admin', value: 'role:admin' },
 glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
 glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
 glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_admin' },
 glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_admin' },
 glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_admin' },
 glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
 glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
 glance-modify_metadef_object: { key: 'modify_metadef_object', value: 'rule:metadef_admin'
},
 glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_admin' },
 glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_admin' },
 glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
 glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
 glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_admin' },
 glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_admin' },
 glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default' },
 glance-get_metadef_properties: { key: 'get_metadef_properties', value: 'rule:metadef_default'
```

```
},
 glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_admin' },
 glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_admin' },
 glance-remove_metadef_property: { key: 'remove_metadef_property', value:
'rule:metadef_admin' },
 glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
 glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
 glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_admin' },
 glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_admin' },
 glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_admin' },
 glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_admin' },
 glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_admin' }
  }

…
```

2. オーバークラウドのデプロイ時に **-e** オプションを使用して、ポリシーオーバーライドが含まれる環境ファイルをデプロイメントコマンドに追加します。

```
$ openstack overcloud deploy -e lock-down-glance-metadef-api.yaml
```

1.1.7.2. metadef API の有効化

以前にメタデータ定義 (metadef) API を制限している場合や、新規のデフォルトを緩和する場合は、metadef 変更ポリシーをオーバーライドして、ユーザーがそれぞれのリソースを更新できるようにすることが可能です。

重要

metadef API への書き込みアクセスに依存するユーザーを管理するクラウド管理者は、すべてのユーザーがこれらの API にアクセスできるようにすることが可能です。ただし、この種の設定では、顧客名や内部プロジェクト等の秘匿すべきリソース名が意図せず漏えいする可能性があります。すべてのユーザーに読み取りアクセスしか付与していない場合であっても、管理者はシステムを監査し、過去に作成したセキュリティー的に脆弱なリソースを識別する必要があります。

手順

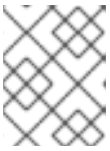
1. クラウド管理者としてアンダークラウドにログインし、ポリシーオーバーライド用のファイルを作成します。以下に例を示します。

```
$ cat open-up-glance-api-metadef.yaml
```

2. すべてのユーザーが metadef API を読み取り/書き込みできるように、ポリシーオーバーライドファイルを設定します。

```
GlanceApiPolicies: {
    glance-metadef_default: { key: 'metadef_default', value: '' },
    glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
    glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
```

```
'rule:metadef_default' },
    glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_default' },
    glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
    glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
    glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_default' },
    glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_default' },
    glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_default'
},
    glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
    glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
    glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default'
},
    glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
    glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_default' },
    glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_default'
},
    glance-remove_metadef_property: { key: 'remove_metadef_property', value:
'rule:metadef_default' },
    glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
    glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
    glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_default' },
    glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_default' },
    glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_default' }
  }
```



注記

すべての metadef ポリシーを設定する際に、**rule:metadeta_default** を使用する必要があります。

3. オーバークラウドのデプロイ時に **-e** オプションを使用して、デプロイメントコマンドに新しいポリシーファイルを追加します。

```
$ openstack overcloud deploy -e open-up-glance-api-metadef.yaml
```

1.2. イメージの管理

Image サービス (glance) は、ディスクおよびサーバーイメージの検出、登録、および配信のサービスを提供します。サーバーイメージのコピーやスナップショットを作成して直ちに保管する機能を提供します。保管したイメージをテンプレートとして使用し、新規サーバーを迅速に稼働させることができま

す。これはサーバーのオペレーティングシステムをインストールしてサービスを個別に設定するよりも一貫性の高い方法です。

1.2.1. イメージの作成

Red Hat Enterprise Linux 7、Red Hat Enterprise Linux 6、または Windows の ISO ファイルを使用して、QCOW2 形式の Red Hat OpenStack Platform (RHOSP) 互換イメージを手動で作成します。

1.2.1.1. Red Hat OpenStack Platform における KVM ゲストイメージの使用

すでに準備済みの RHEL KVM ゲスト QCOW2 イメージを使用することができます。

- Red Hat Enterprise Linux 8 KVM Guest Image
- Red Hat Enterprise Linux 7 KVM Guest Image
- Red Hat Enterprise Linux 6 KVM Guest Image

これらのイメージは、**cloud-init** を使用して設定されます。適切に機能させるには、ec2 互換のメタデータサービスを利用して SSH キーをプロビジョニングする必要があります。

準備済みの Windows KVM ゲスト QCOW2 イメージはありません。

注記

KVM ゲストイメージでは、以下の点に注意してください。

- KVM ゲストイメージでは **root** アカウントが無効になっていますが、**cloud-user** という名前の特別なユーザーに **sudo** アクセスが許可されています。
- このイメージには **root** パスワードは設定されていません。

root パスワードは、**/etc/shadow** で 2 番目のフィールドに **!!** と記載することによりロックされます。

RHOSP インスタンスでは、RHOSP Dashboard またはコマンドラインから ssh キーペアを生成し、その鍵の組み合わせを使用して、インスタンスに対して root として SSH 公開認証を実行します

インスタンスの起動時には、この公開鍵がインスタンスに挿入されます。続いて、キーペア作成時にダウンロードする秘密鍵を使用して認証を行うことができます。

Red Hat Enterprise Linux または Windows のカスタムイメージを作成する場合は、Red Hat Enterprise Linux 7 イメージの作成、Red Hat Enterprise Linux 6 イメージの作成 、または Windows イメージの作成 を参照してください。

1.2.1.2. Red Hat Enterprise Linux または Windows のカスタムイメージの作成

前提条件

- イメージを作成する Linux ホストマシン。これは、アンダークラウドまたはオーバークラウドを除いて、Linux パッケージをインストールして実行することのできる任意のマシンです。
- advanced-virt リポジトリーが有効になっています。

```
$ sudo subscription-manager repos --enable=advanced-virt-for-rhel-8-x86_64-rpms
```

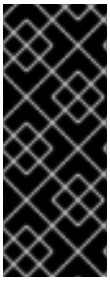
- ゲストオペレーティングシステムを作成するのに必要なすべてのパッケージをインストールする libvirt、virt-manager

```
$ sudo dnf module install -y virt
```

- 仮想マシンイメージにアクセスして変更するための一連のツールをインストールする Libguestfs ツール

```
$ sudo dnf install -y libguestfs-tools-c
```

- Red Hat Enterprise Linux 7 もしくは 6 の ISO ファイル(詳細については、RHEL 7.2 Binary DVD もしくは RHEL 6.8 Binary DVD を参照) または Windows の ISO ファイル。Windows の ISO ファイルがない場合には、Microsoft TechNet Evaluation Center に移動して評価版イメージをダウンロードしてください。
- **キックスタート** ファイルを編集する必要がある場合にはテキストエディター (RHEL のみ)



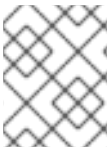
重要

アンダークラウドに **libguestfs-tools** パッケージをインストールする場合は、アンダークラウドの **tripleo_iscsid** サービスとのポートの競合を避けるために **iscsid.socket** を無効にします。

```
$ sudo systemctl disable --now iscsid.socket
```

1.2.1.2.1. Red Hat Enterprise Linux 7 イメージの作成

Red Hat Enterprise Linux 7 の ISO ファイルを使用して、QCOW2 形式の Red Hat OpenStack Platform (RHOSP) 互換イメージを手動で作成します。



注記

[root@host]# プロンプトのすべてのコマンドを、ホストマシン上で実行する必要があります。

1. **virt-install** でインストールを開始します。

```
[root@host]# qemu-img create -f qcow2 rhel7.qcow2 8G
[root@host]# virt-install --virt-type kvm --name rhel7 --ram 2048 \
--cdrom /tmp/rhel-server-7.2-x86_64-dvd.iso \
--disk rhel7.qcow2,format=qcow2 \
--network=bridge:virbr0 --graphics vnc,listen=0.0.0.0 \
--noautoconsole --os-type=linux --os-variant=rhel7
```

このコマンドによりインスタンスが起動し、インストールプロセスが開始されます。

注記

インスタンスが自動的に起動しない場合には、**virt-viewer** のコマンドを実行して、コンソールを確認します。

```
[root@host]# virt-viewer rhel7
```

2. インスタンスを設定します。

   a. インストーラーの初期ブートメニューで、**Install Red Hat Enterprise Linux 7**を選択します。

   b. 適切な **言語** および **キーボード** オプションを選択します。

   c. インストールに使用するデバイス種別を尋ねるプロンプトが表示されたら、**自動検出したインストールメディア** を選択します。

   d. インストール先を尋ねるプロンプトが表示されたら、**ローカルの標準ディスク** を選択します。その他のストレージタイプオプションには、**自動設定のパーティション設定** を選択します。

   e. ソフトウェアのオプションには、**最小限のインストール** を選択します。

   f. ネットワークとホスト名の設定では、ネットワークに **eth0** を選択し、デバイスのホスト名を指定します。デフォルトのホスト名は **localhost.localdomain** です。

   g. **root パスワード** フィールドにパスワードを入力し、**確認** フィールドに同じパスワードをもう一度入力します。

   結果

   インストールプロセスが完了すると、**完了しました!** の画面が表示されます。

3. インストールが完了した後には、インスタンスを再起動して、root ユーザーとしてログインします。

4. **/etc/sysconfig/network-scripts/ifcfg-eth0** ファイルを編集して、以下の値のみが記載されている状態にします。

```
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

5. マシンを再起動します。

6. コンテンツ配信ネットワークにマシンを登録します。

```
# sudo subscription-manager register
# sudo subscription-manager attach --pool=Valid-Pool-Number-123456
# sudo subscription-manager repos --enable=rhel-7-server-rpms
```

7. システムを更新します。

```
# dnf -y update
```

8. **cloud-init** パッケージをインストールします。

```
# dnf install -y cloud-utils-growpart cloud-init
```

9. **/etc/cloud/cloud.cfg** 設定ファイルを編集して、**cloud_init_modules** の下に以下を追加します。

```
- resolv-conf
```

resolv-conf オプションは、インスタンスの初回起動時に **resolv.conf** を自動的に設定します。このファイルには、**nameservers**、**domain**、その他のオプションなどのインスタンスに関連した情報が記載されています。

10. **/etc/sysconfig/network** に以下の行を追加し、EC2 メタデータサービスへのアクセスで問題が発生するのを回避します。

```
NOZEROCONF=yes
```

11. コンソールメッセージが Dashboard の **ログ** タブおよび **nova console-log** の出力に表示されるようにするには、以下のブートオプションを **/etc/default/grub** ファイルに追記します。

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
```

12. **grub2-mkconfig** コマンドを実行します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

以下のような出力が表示されます。

```
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-229.7.2.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-229.7.2.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-121.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-121.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-b82a3044fb384a3f9aeacf883474428b
Found initrd image: /boot/initramfs-0-rescue-b82a3044fb384a3f9aeacf883474428b.img
done
```

13. インスタンスの登録を解除して、作成されるイメージにこのインスタンスのサブスクリプション情報が含まれないようにします。

```
# subscription-manager repos --disable=*
# subscription-manager unregister
# dnf clean all
```

14. インスタンスの電源をオフにします。

```
# poweroff
```

15. **virt-sysprep** コマンドでイメージのリセットおよびクリーニングをして、問題なくインスタンスの作成に使用できるようにします。

```
[root@host]# virt-sysprep -d rhel7
```

16. ディスクイメージ内の空き容量をホスト内の空き容量に戻して、イメージサイズを縮小します。

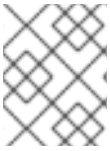
```
[root@host]# virt-sparsify --compress /tmp/rhel7.qcow2 rhel7-cloud.qcow2
```

このコマンドを実行すると、その場所に **rhel7-cloud.qcow2** ファイルが作成されます。

rhel7-cloud.qcow2 イメージファイルを Image サービスにアップロードする準備が整いました。Dashboard を使用してこのイメージを RHOSP デプロイメントにアップロードする方法については、イメージのアップロード を参照してください。

1.2.1.2.2. Red Hat Enterprise Linux 6 イメージの作成

Red Hat Enterprise Linux 6 の ISO ファイルを使用して、QCOW2 形式の Red Hat OpenStack Platform (RHOSP) 互換イメージを手動で作成します。



注記

[root@host]# プロンプトのすべてのコマンドを、ホストマシン上で実行する必要があります。

1. **virt-install** でインストールを開始します。

```
[root@host]# qemu-img create -f qcow2 rhel6.qcow2 4G
[root@host]# virt-install --connect=qemu:///system --network=bridge:virbr0 \
--name=rhel6 --os-type linux --os-variant rhel6 \
--disk path=rhel6.qcow2,format=qcow2,size=10,cache=none \
--ram 4096 --vcpus=2 --check-cpu --accelerate \
--hvm --cdrom=rhel-server-6.8-x86_64-dvd.iso
```

このコマンドによりインスタンスが起動し、インストールプロセスが開始されます。

注記

インスタンスが自動的に起動しない場合には、**virt-viewer** のコマンドを実行して、コンソールを確認します。

```
[root@host]# virt-viewer rhel6
```

2. インスタンスを設定します。

   a. インストーラーの初期ブートメニューで **Install or upgrade an existing system**を選択し、インストールの指示に従います。デフォルト値を受け入れます。
   ディスクインストーラーでは、インストール前にインストールメディアをテストするオプションを利用することができます。テストを実行するには **OK** を、テストを行わずに続行するには **Skip** を選択します。

   b. 適切な **言語** および **キーボード** オプションを選択します。

   c. インストールに使用するデバイス種別を尋ねるプロンプトが表示されたら、**基本ストレージデバイス** を選択します。

d. デバイスのホスト名を指定します。デフォルトのホスト名は **localhost.localdomain** です。

e. **タイムゾーン** と **root** パスワードを設定します。

f. **Which type of installation would you like?**のウィンドウのオプションから、ディスクの空き容量に応じて必要なインストールの種別を選択します。

g. SSH サーバーをインストールする **基本サーバー** インストールを選択します。

h. インストールプロセスが完了し、**おめでとうございます。Red Hat Enterprise Linux のインストールが完了しました。**の画面が表示されます。

3. インスタンスを再起動して、**root** ユーザーとしてログインします。

4. **/etc/sysconfig/network-scripts/ifcfg-eth0** ファイルを編集して、以下の値のみが記載されている状態にします。

```
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

5. マシンを再起動します。

6. コンテンツ配信ネットワークにマシンを登録します。

```
# sudo subscription-manager register
# sudo subscription-manager attach --pool=Valid-Pool-Number-123456
# sudo subscription-manager repos --enable=rhel-6-server-rpms
```

7. システムを更新します。

```
# dnf -y update
```

8. **cloud-init** パッケージをインストールします。

```
# dnf install -y cloud-utils-growpart cloud-init
```

9. **/etc/cloud/cloud.cfg** 設定ファイルを編集し、**cloud_init_modules** の下に以下の内容を追加します。

```
- resolv-conf
```

resolv-conf オプションは、インスタンスの初回起動時に **resolv.conf** 設定ファイルを自動的に設定します。このファイルには、**nameservers**、**domain**、その他のオプションなどのインスタンスに関連した情報が記載されています。

10. ネットワークの問題が発生するのを防ぐために、**/etc/udev/rules.d/75-persistent-net-generator.rules** ファイルを作成します。

```
# echo "#" > /etc/udev/rules.d/75-persistent-net-generator.rules
```

これにより、**/etc/udev/rules.d/70-persistent-net.rules** ファイルが作成されるのを防ぎま

す。**/etc/udev/rules.d/70-persistent-net.rules** が作成されてしまうと、スナップショットからのブート時にネットワークが適切に機能しなくなる可能性があります (ネットワークインターフェイスが **eth0** ではなく **eth1** として作成され、IP アドレスが割り当てられません)。

11. **/etc/sysconfig/network** に以下の行を追加し、EC2 メタデータサービスへのアクセスで問題が発生するのを回避します。

```
NOZEROCONF=yes
```

12. コンソールメッセージが Dashboard の **ログ** タブおよび **nova console-log** の出力に表示されるようにするには、以下のブートオプションを **/etc/grub.conf** ファイルに追記します。

```
console=tty0 console=ttyS0,115200n8
```

13. 仮想マシンの登録を解除して、作成されるイメージにこのインスタンスと同じサブスクリプション情報が含まれないようにします。

```
# subscription-manager repos --disable=*
# subscription-manager unregister
# dnf clean all
```

14. インスタンスの電源をオフにします。

```
# poweroff
```

15. **virt-sysprep** コマンドでイメージのリセットおよびクリーニングをして、問題なくインスタンスの作成に使用できるようにします。

```
[root@host]# virt-sysprep -d rhel6
```

16. **virt-sparsify** コマンドを使用してイメージのサイズを縮小します。このコマンドにより、ディスクイメージ内の空き容量は、ホスト内の空き容量に戻ります。

```
[root@host]# virt-sparsify --compress rhel6.qcow2 rhel6-cloud.qcow2
```

このコマンドを実行すると、その場所に新しい **rhel6-cloud.qcow2** ファイルが作成されます。

注記

インスタンスに適用されているフレーバーのディスクスペースに応じて、イメージをベースとするインスタンスのパーティションを手動でリサイズする必要があります。

rhel6-cloud.qcow2 イメージファイルを Image サービスにアップロードする準備が整いました。Dashboard を使用してこのイメージを RHOSP デプロイメントにアップロードする方法については、イメージのアップロード を参照してください。

1.2.1.2.3. Windows イメージの作成

Windows の ISO ファイルを使用して、QCOW2 形式の Red Hat OpenStack Platform (RHOSP) 互換イメージを手動で作成します。

注記

[root@host]# プロンプトのすべてのコマンドを、ホストマシン上で実行する必要があります。

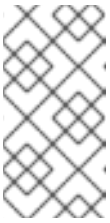
手順

1. **virt-install** でインストールを開始します。

```
[root@host]# virt-install --name=<name> \
--disk size=<size> \
--cdrom=<path> \
--os-type=windows \
--network=bridge:virbr0 \
--graphics spice \
--ram=<ram>
```

virt-install パラメーターの以下の値を置き換えます。

- <name>: Windows インスタンスの名前
- size: ディスクのサイズ (GB)
- path: Windows のインストール ISO ファイルへのパス
- RAM: 要求するメモリー容量 (MB)



注記

--os-type=windows パラメーターにより、Windows ゲストのクロックが正しく設定され、Hyper-V エンライトメント機能が有効化されるようになります。Image サービスにイメージをアップロードする前に、イメージメタデータに **os_type=windows** を設定する必要もあります。

2. デフォルトでは、**virt-install** は **/var/lib/libvirt/images/<name>.qcow2** としてゲストイメージを保存します。ゲストイメージを別の場所に保存する場合は、**--disk** オプションのパラメーターを変更します。

```
--disk path=<filename>,size=<size>
```

<filename> を、インスタンスイメージを保存するファイルの名前 (およびオプションでそのパス) に置き換えます。たとえば、**path=win8.qcow2,size=8** は現在の作業ディレクトリーに **win8.qcow2** という名前の 8 GB ファイルを作成します。

ヒント

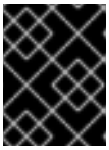
ゲストが自動的に起動しない場合には、**virt-viewer** のコマンドを実行して、コンソールを確認します。

```
[root@host]# virt-viewer <name>
```

Windows のインストール方法に関する詳細は、該当する Microsoft のドキュメントを参照してください。

3. 新規インストールした Windows システムで仮想化ハードウェアを使用できるようにするには、virtio ドライバーをインストールしなければならない場合があります。そのためには、まずイメージをインストールし、それを CD-ROM ドライブとして Windows インスタンスにアタッチする必要があります。**virtio-win** パッケージをインストールするには、インスタンスに virtio ISO イメージを追加して、virtio ドライバーをインストールする必要があります。詳細については、**仮想化の設定および管理** の Windows 仮想マシン用の KVM 準仮想化ドライバーのインストール を参照してください。

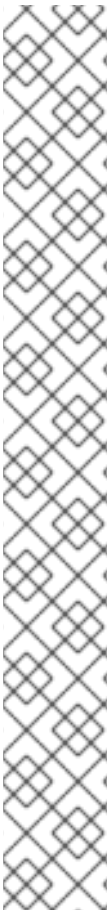
4. Windows システムで Cloudbase-Init をダウンロード、実行して、設定を完了します。Cloudbase-Init のインストールの最後に、**Run Sysprep** と **Shutdown** チェックボックスを選択します。**Sysprep** ツールは、特定の Microsoft サービスで使用する OS ID を生成して、ゲストを一意にします。



重要

Red Hat は Cloudbase-Init に関するテクニカルサポートは提供しません。問題が発生した場合は、Cloudbase Solutions にお問い合わせください。

Windows システムがシャットダウンしたら、**<name>.qcow2** イメージファイルを Image サービスにアップロードすることができます。Dashboard またはコマンドラインを使用してこのイメージを RHOSP デプロイメントにアップロードする方法については、イメージのアップロード を参照してください。



注記

libosinfo データ

Compute サービスでは、libosinfo データを使用してデフォルトのデバイスモデルを設定する操作が非推奨になりました。これに代わって、以下のイメージメタデータ属性を使用して、インスタンス用の最適な仮想ハードウェアを設定します。

- **os_distro**
- **os_version**
- **hw_cdrom_bus**
- **hw_disk_bus**
- **hw_scsi_model**
- **hw_vif_model**
- **hw_video_model**
- **hypervisor_type**

これらのメタデータ属性についての詳細は、付録A *イメージの設定パラメーター*を参照してください。

1.2.2. イメージのアップロード

1. Dashboard で **プロジェクト > コンピュート > イメージ**を選択します。
2. **イメージの作成** をクリックします。

3. 各フィールドに値を入力し、**イメージの作成** をクリックします。

表1.1 イメージのオプション

フィールド	説明
名前	イメージの名前。そのプロジェクト内で一意な名前にする必要があります。
説明	イメージを識別するための簡単な説明
イメージソース	イメージソース: **イメージの場所** または **イメージファイル**。ここで選択したオプションに応じて次のフィールドが表示されます。
イメージの場所またはイメージファイル	• イメージの場所の URL を指定するには、**イメージの場所** オプションを選択します。 • ローカルディスクからイメージをアップロードするには、**イメージファイル** オプションを選択します。
形式	イメージの形式 (例: qcow2)
アーキテクチャー	イメージのアーキテクチャー。たとえば 32 ビットのアーキテクチャーには i686、64 ビットのアーキテクチャーには x86_64 を使用します。
最小ディスク (GB)	イメージのブートに必要な最小のディスクサイズ。このフィールドに値が指定されていない場合には、デフォルト値は 0 です (最小値なし)。
最小メモリー (MB)	イメージのブートに必要な最小のメモリーサイズ。このフィールドに値が指定されていない場合には、デフォルト値は 0 です (最小値なし)。
パブリック	このチェックボックスを選択した場合には、プロジェクトにアクセスできる全ユーザーにイメージが公開されます。
保護	このチェックボックスを選択した場合には、特定のパーミッションのあるユーザーのみがこのイメージを削除できるようになります。

イメージが正常にアップロードされるとそのステータスが **active** になり、イメージが使用可能であることを示します。Image サービスは、アップロードの開始時に使用した Identity サービストークンのライフタイムよりもアップロードの所要時間が長くかかる大容量のイメージも処理することができる点に注意してください。これは、アップロードが完了してイメージのステータスが更新される際に、新しいトークンを取得して使用できるように、Identity サービスは最初に Identity サービスとのトラストを作成するためです。

注記

glance image-create コマンドに **property** のオプションを指定して実行する方法でイメージをアップロードすることもできます。コマンドラインで操作を行った方が、より多くの値を使用することができます。完全なリストは、イメージの設定パラメーター を参照してください。

1.2.3. イメージの更新

1. Dashboard で **プロジェクト > コンピュート > イメージ**を選択します。
2. リストから **イメージの編集** をクリックします。

注記

イメージの編集 オプションは、**admin** ユーザーとしてログインした場合にのみ使用することができます。**demo** ユーザーとしてログインした場合には、**インスタンスの起動** または **ボリュームの作成** のオプションを使用することができます。

3. フィールドを更新し、**イメージの更新** をクリックします。次の値を更新することができます (名前、説明、カーネル ID、RAM ディスク ID、アーキテクチャー、形式、最小ディスク、最小メモリー、パブリック、保護)。
4. ドロップダウンメニューをクリックして **メタデータの更新** オプションを選択します。
5. 左のコラムから右のコラムに項目を追加して、メタデータを指定します。左のコラムには、Image サービスのメタデータカタログからのメタデータの定義が含まれています。**その他** を選択して、任意のキーを使用してメタデータを追加し、完了したら **保存** をクリックします。

注記

glance image-update コマンドに **property** オプションを指定して実行する方法でイメージを更新することもできます。コマンドラインで操作を行った方が、より多くの値を使用することができます。完全なリストは、イメージの設定パラメーター を参照してください。

1.2.4. イメージのインポート

次の 2 つの方法のいずれかを使用して、Image サービス (glance) にイメージをインポートできます。

- **web-download** を使用して、URI からイメージをインポートします。
- ローカルファイルシステムからイメージをインポートするには、**glance-direct** を使用します。

web-download メソッドはデフォルトで有効化されています。クラウド管理者がインポート方法を設定します。利用可能なインポートオプションを一覧表示するには、**glance import-info** コマンドを実行します。

1.2.4.1. リモート URI からのイメージのインポート

web-download メソッドを使用して、リモートの URI からイメージをコピーすることができます。

1. イメージを作成して、インポートするイメージの URI を指定します。

```
$ glance image-create-via-import \
   --container-format <CONTAINER FORMAT> \
   --disk-format <DISK-FORMAT> \
   --name <NAME> \
   --import-method web-download \
   --uri <URI>
```

- **<CONTAINER FORMAT>** は、イメージに設定するコンテナー形式 (None、ami、ari、aki、bare、ovf、ova、docker) に置き換えます。
- **<DISK-FORMAT>** は、イメージに設定しているディスクフォーマット (None、ami、ari、aki、vhd、vhdx、vmdk、raw、qcow2、vdi、iso、ploop) に置き換えます。
- **<NAME>** は、イメージのわかりやすい名前に置き換えます。
- **<URI>** は、イメージの URI に置き換えます。

2. **glance image-show <IMAGE_ID>** コマンドを使用して、イメージの可用性を確認できます。
   - **<IMAGE_ID>** は、イメージの作成時に指定した ID に置き換えます。

Image サービスの **web download** メソッドでは、2 段階のプロセスでインポートを実施します。

1. **Web ダウンロード** 方式では、イメージレコードが作成されます。
2. **Web ダウンロード** メソッドは、指定された URI からイメージを取得します。

URI は、オプションの拒否リストおよび許可リストのフィルタリングの対象となります。

Image Property Injection プラグインにより、メタデータ属性をイメージに注入することができます。注入されたこれらの属性により、イメージインスタンスを起動するコンピュートノードが決定されます。

1.2.4.2. ローカルボリュームからのイメージのインポート

glance-direct メソッドは、イメージレコードを作成し、それによりイメージ ID が生成されます。イメージがローカルボリュームから Image サービスにアップロードされるとステージングエリアに保管され、設定されているチェックに合格した後にアクティブとなります。高可用性 (HA) 設定で使用される場合には、**glance-direct** メソッドには共通のステージングエリアが必要です。

注記

HA 環境では、**glance-direct** メソッドを使用したイメージのアップロードは、共通のステージエリアがない場合には失敗します。HA の アクティブ/アクティブ環境では、API コールは複数の Image サービスのコントローラーに分散されます。ダウンロード API コールは、イメージをアップロードする API コールとは別のコントローラーに送信することが可能です。

glance-direct メソッドは、3 つの異なるコールを使用して、イメージをインポートします。

- **glance image-create**
- **glance image-stage**
- **glance image-import**

glance image-create-via-import コマンドを使用すると、これらの 3 つのコールを 1 つのコマンドで実行することができます。

```
$ glance image-create-via-import --container-format <CONTAINER FORMAT> --disk-format <DISK-FORMAT> --name <NAME> --file </PATH/TO/IMAGE>
```

- **<CONTAINER FORMAT>**、**<DISK-FORMAT>**、**<NAME>**、および **</PATH/TO/IMAGE>** は、イメージに関連する値に置き換えます。

イメージがステージングエリアからバックエンドの場所に移動すると、そのイメージはリストされます。ただし、イメージがアクティブになるには、多少時間がかかる場合があります。

glance image-show <IMAGE_ID> コマンドを使用して、イメージの可用性を確認できます。

- **<IMAGE_ID** は、イメージの作成時に指定した ID に置き換えます。

1.2.5. イメージを削除します。

1. Dashboard で **プロジェクト > コンピュート > イメージ**を選択します。
2. 削除するイメージを選択し、**イメージの削除** ボタンをクリックします。

1.2.6. イメージの表示または非表示

ユーザーに表示される通常のリストからパブリックイメージを非表示にすることができます。たとえば、廃止された CentOS 7 イメージを非表示にし、最新バージョンだけを表示してユーザーエクスペリエンスをシンプル化することができます。ユーザーは、非表示のイメージを検出して使用することができます。

イメージを非表示にするには、以下をコマンドを実行します。

```
glance image-update <image_id> --hidden 'true'
```

非表示のイメージを作成するには、**glance image-create** コマンドに **--hidden** 引数を追加します。

イメージの非表示を解除するには、以下のコマンドを実行します。

```
glance image-update <image_id> --hidden 'false'
```

1.2.7. 非表示にしたイメージの表示

非表示にしたイメージをリスト表示するには、以下のコマンドを実行します。

```
glance image-list --hidden 'true'
```

1.2.8. イメージ変換の有効化

GlanceImageImportPlugins パラメーターを有効にすると、QCOW2 イメージをアップロードすることができ、Image サービスはそのイメージを RAW に変換します。

注記

Red Hat Ceph Storage RBD を使用してイメージを保存して Nova インスタンスをブートすると、イメージの変換は自動的に有効になります。

イメージの変換を有効にするには、以下のパラメーター値が含まれる環境ファイルを作成し、**-e** オプションを使用して新しい環境ファイルを **openstack overcloud deploy** コマンドに追加します。

```
parameter_defaults:
  GlanceImageImportPlugins:'image_conversion'
```

1.2.9. RAW 形式へのイメージの変換

Red Hat Ceph Storage は QCOW2 イメージを保管することはできますが、そのイメージを使用して仮想マシン (VM) のディスクをホストすることはできません。

アップロードした QCOW2 イメージから仮想マシンを作成する場合には、コンピュートノードはイメージをダウンロードして RAW に変換し、それを Ceph にアップロードし直してからでないと使用することができません。このプロセスは仮想マシンの作成時間に影響を及ぼします (特に、並行して仮想マシンを作成する場合)。

たとえば、複数の仮想マシンを同時に作成する場合には、Ceph クラスターへの変換済みイメージのアップロードが、すでに実行中の負荷に影響を及ぼす可能性があります。IOPS のこれらの負荷に対するリソースがアップロードプロセスにより枯渇し、ストレージの反応が遅くなる場合があります。

Ceph において仮想マシンをより効率的に起動するには (一時バックエンドまたはボリュームからの起動)、glance イメージの形式を RAW にする必要があります。

手順

1. イメージを RAW に変換すると、イメージサイズが元の QCOW2 イメージファイルより大きくなる場合があります。最終的な RAW イメージのサイズを確認するには、変換前に以下のコマンドを実行します。

```
qemu-img info <image>.qcow2
```

2. イメージを QCOW2 から RAW 形式に変換します。

```
qemu-img convert -p -f qcow2 -O raw <original qcow2 image>.qcow2 <new raw image>.raw
```

1.2.9.1. Image サービス (glance) でのディスクフォーマットの設定

GlanceDiskFormats パラメーターを使用して、ディスクフォーマットを有効または拒否するように Image サービス (glance) を設定することができます。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. 環境ファイルに **GlanceDiskFormats** パラメーターを追加します (例: **glance_disk_formats.yaml**)。

```
parameter_defaults:
  GlanceDiskFormats:
    - <disk_format>
```

- たとえば、RAW および ISO ディスクフォーマットだけを有効にするには、以下の設定を使用します。

```
parameter_defaults:
  GlanceDiskFormats:
```

```
    - raw
    - iso
```

- QCOW2 ディスクイメージを拒否するには、以下の設定例を使用します。

```
parameter_defaults:
  GlanceDiskFormats:
  - raw
  - iso
  - aki
  - ari
  - ami
```

4. ご自分の環境に該当するその他の環境ファイルと共に、新しい設定が含まれる環境ファイルを **openstack overcloud deploy** コマンドに追加します。

```
$ openstack overcloud deploy --templates \
  -e <overcloud_environment_files> \
  -e <new_environment_file> \
  …
```

- **<overcloud_environment_files>** をデプロイメントに追加する環境ファイルのリストに置き換えます。
- **<new_environment_file>** を新しい設定が含まれる環境ファイルに置き換えます。

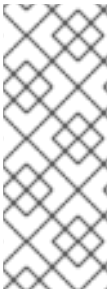
RHOSP で利用可能なディスクフォーマットの詳細は、Image サービス を参照してください。

1.2.10. RAW 形式でのイメージの保存

以前に作成したイメージを RAW 形式で保存するには、**GlanceImageImportPlugins** パラメーターを有効にして以下のコマンドを実行します。

```
$ glance image-create-via-import \
    --disk-format qcow2 \
    --container-format bare \
    --name NAME \
    --visibility public \
    --import-method web-download \
    --uri http://server/image.qcow2
```

- **--name**: **NAME** をイメージ名に置き換えます。この名前が **glance image-list** に表示されます。
- **--uri**: **http://server/image.qcow2** を QCOW2 イメージの場所およびファイル名に置き換えます。



注記

このコマンド例では、イメージレコードを作成し、**web-download** メソッドを使用してそのイメージレコードをインポートします。glance-api は、インポートプロセス中に **--uri** で定義した場所からイメージをダウンロードします。**web-download** が利用できない場合、**glanceclient** はイメージデータを自動的にダウンロードすることができません。利用可能なイメージのインポート方法をリスト表示するには、**glance import-info** コマンドを実行します。

第2章 複数のストアに対応した IMAGE サービス

Red Hat OpenStack Platform Image サービス (glance) では、分散エッジアーキテクチャーによる複数ストアの使用がサポートされます。そのため、すべてのエッジサイトにイメージプールを設定することができます。ハブサイトとも呼ばれる中央サイトとエッジサイトの間で、イメージをコピーすることができます。

イメージのメタデータには、各コピーの場所が含まれます。たとえば、2 つのエッジサイトに存在するイメージは、3 つの場所 (中央サイトおよび 2 つのエッジサイト) に単一の UUID として公開されます。つまり、多くのストアで単一の UUID を共有するイメージデータのコピーを持つことができます。場所についての詳細は、イメージの場所について を参照してください。

すべてのエッジサイトに RBD イメージプールがあるため、Ceph RBD の Copy-on-Write (COW) およびスナップショットの階層化技術を使用して仮想マシンを迅速にブートすることができます。これは、仮想マシンをボリュームからブートできるのと共に、ライブマイグレーションが可能であることを意味します。Ceph RBD を使用した階層化についての詳細は、**ブロックデバイスガイド**の Ceph ブロックデバイスの階層化 を参照してください。

2.1. ストレージエッジアーキテクチャーの要件

- 各イメージのコピーは、中央サイトの Image サービスに存在している必要があります。
- エッジサイトでインスタンスを作成する前に、そのエッジサイトにイメージのローカルコピーが必要です。
- あるエッジサイトにアップロードしたイメージを他のエッジサイトにコピーする前に、そのイメージを中央サイトにコピーする必要があります。
- すべてのエッジサイトで Image サービス RBD ドライバーを使用します。混合アーキテクチャーはサポートされません。
- RBD は、Image、Compute、および Block Storage サービスのストレージドライバーでなければなりません。
- それぞれのサイトで、**NovaComputeAvailabilityZone** および **CinderStorageAvailabilityZone** パラメーターに同じ値を割り当てる必要があります。

2.2. 複数ストアへのイメージのインポート

相互運用可能なイメージのインポートワークフローを使用して、イメージデータを複数の Ceph Storage クラスターにインポートします。ローカルファイルシステムで、または Web サーバーから利用可能なイメージを、Image サービスにインポートすることができます。

Web サーバーからイメージをインポートする場合、イメージを複数のストアに一度にインポートすることができます。イメージが Web サーバーで利用できない場合は、イメージをローカルファイルシステムから中央のストアにインポートし、それをさらに別のストアにコピーすることができます。詳しくは、複数ストアへの既存イメージのコピー を参照してください。

重要

中央サイトにイメージを使用するインスタンスがない場合でも、必ず中央サイトにイメージのコピーを保存してください。Image サービスへのイメージのインポートについての詳しい情報は、Distributed compute node and storage deployment を参照してください。

2.2.1. イメージのインポート失敗時の対応

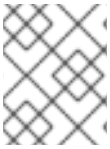
--allow-failure パラメーターを使用して、イメージインポート操作の失敗に対応することができます。

- **--allow-failure** パラメーターの値を **true** に設定した場合、最初のストアにデータが正常にインポートされると、イメージのステータスは **active** になります。これがデフォルトの設定です。**os_glance_failed_import** イメージ属性を使用して、イメージデータのインポートに失敗したストアのリストを表示することができます。

- **--allow-failure** パラメーターの値を **false** に設定すると、指定したすべてのストアにデータが正常にインポートされた場合に限り、イメージのステータスが **active** になります。いずれかのストアでイメージデータのインポートが失敗した場合、イメージのステータスは **failed** になります。イメージは指定されたどのストアにもインポートされません。

2.2.2. 複数ストアへのイメージデータのインポート

--allow-failure パラメーターのデフォルト設定は **true** なので、一部のストアがイメージデータのインポートに失敗するのを許容するのであれば、コマンドにパラメーターを追加する必要はありません。



注記

この手順では、すべてのストアがイメージデータを正常にインポートすることは求められません。

手順

1. 指定した複数のストアにイメージデータをインポートします。

```
$ glance image-create-via-import \
--container-format bare \
--name IMAGE-NAME \
--import-method web-download \
--uri URI \
--stores STORE1,STORE2,STORE3
```

   - **IMAGE-NAME** をインポートするイメージの名前に置き換えます。
   - **URI** をイメージの URI に置き換えます。
   - **STORE1**、**STORE2**、および **STORE3** を、イメージデータをインポートするストアの名前に置き換えます。
   - あるいは、**--stores** を **--all-stores true** に置き換え、すべてのストアにイメージをアップロードします。

注記

QCOW2 イメージを自動的に RAW 形式に変換する **glance image-create-via-import** コマンドは、**web-download** メソッドでのみ機能します。**glance-direct** メソッドを使用することはできますが、共有ファイルシステムが設定されたデプロイメントでのみ機能します。詳細は、RAW 形式でのイメージの保存 を参照してください。

2.2.3. 複数ストアへのイメージデータのインポート (失敗を許容しない)

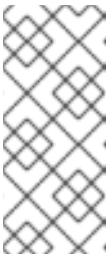
この手順では、すべてのストアがイメージデータを正常にインポートすることが求められます。

手順

1. 指定した複数のストアにイメージデータをインポートします。

```
$ glance image-create-via-import \
--container-format bare \
--name IMAGE-NAME \
--import-method web-download \
--uri URI \
--stores STORE1,STORE2
```

- **IMAGE-NAME** をインポートするイメージの名前に置き換えます。
- **URI** をイメージの URI に置き換えます。
- **STORE1**、**STORE2**、および **STORE3** を、イメージデータをコピーするストアの名前に置き換えます。
- あるいは、**--stores** を **--all-stores true** に置き換え、すべてのストアにイメージをアップロードします。



注記

--allow-failure パラメーターを **false** に設定すると、Image サービスはイメージデータのインポートに失敗したストアを無視しません。イメージ属性 **os_glance_failed_import** を使用して、失敗したストアのリストを表示することができます。詳細は、イメージインポート操作の進捗の確認 を参照してください。

2. イメージデータが特定のストアに追加されたことを確認します。

```
$ glance image-show IMAGE-ID | grep stores
```

IMAGE-ID を元の既存イメージの ID に置き換えます。

出力には、ストアのコンマ区切りリストが表示されます。

2.2.4. 1 つのストアへのイメージデータのインポート

イメージデータを 1 つのストアにインポートすることができます。

手順

1. イメージデータを 1 つのストアにインポートします。

```
$ glance image-create-via-import \
--container-format bare \
--name IMAGE-NAME \
--import-method web-download \
--uri URI \
--store STORE
```

- **IMAGE-NAME** をインポートするイメージの名前に置き換えます。

- **URI** をイメージの URI に置き換えます。
- **STORE** をイメージデータをコピーするストアの名前に置き換えます。

注記

コマンドに **--stores**、**--all-stores**、または **--store** オプションを指定しないと、Image サービスは中央ストアにイメージを作成します。

2. イメージデータが特定のストアに追加されたことを確認します。

```
$ glance image-show IMAGE-ID | grep stores
```

IMAGE-ID を元の既存イメージの ID に置き換えます。

出力には、ストアのコンマ区切りリストが表示されます。

2.2.5. イメージインポート操作の進捗の確認

相互運用可能なイメージのインポートワークフローでは、イメージデータが順次ストアにインポートされます。イメージのサイズ、ストア数、および中央サイトとエッジサイト間のネットワーク速度が、イメージのインポート操作が完了するのにかかる時間に影響を及ぼします。

イメージのインポート操作中に送付される通知に表示される 2 つのイメージ属性を見て、イメージインポートの進捗を把握することができます。

- **os_glance_importing_to_stores** 属性: イメージデータをインポートしていないストアがリスト表示されます。インポートの開始時点では、要求されたすべてのストアがリストに表示されます。ストアがイメージデータを正常にインポートするたびに、Image サービスはストアをリストから削除します。
- **os_glance_failed_import** 属性: イメージデータのインポートに失敗したストアがリスト表示されます。イメージインポート操作の開始時点では、このリストには何も表示されません。

注記

以下の手順の環境には、**central** ストアおよび 2 つのエッジストア (**dcn0** および **dcn1**) という 3 つの Ceph Storage クラスターがあります。

手順

1. イメージデータが特定のストアに追加されたことを確認します。

```
$ glance image-show IMAGE-ID
```

IMAGE-ID を元の既存イメージの ID に置き換えます。

出力には、以下のスニペット例のようなストアのコンマ区切りリストが表示されます。

```
| os_glance_failed_import       |
| os_glance_importing_to_stores | central,dcn0,dcn1
| status                        | importing
```

2. イメージインポート操作のステータスを監視します。このコマンドを **watch** コマンドの引数にすると、コマンドの出力は 2 秒ごとに更新されます。

```
$ watch glance image-show IMAGE-ID
```

IMAGE-ID を元の既存イメージの ID に置き換えます。

イメージのインポート操作が進むと、操作のステータスが変わります。

```
| os_glance_failed_import      |
| os_glance_importing_to_stores | dcn0,dcn1
| status                       | importing
```

イメージのインポートに失敗したことを示す出力は、以下の例のようになります。

```
| os_glance_failed_import      | dcn0
| os_glance_importing_to_stores | dcn1
| status                       | importing
```

操作が完了すると、ステータスが active に変わります。

```
| os_glance_failed_import      | dcn0
| os_glance_importing_to_stores |
| status                       | active
```

2.3. 複数ストアへの既存イメージのコピー

この機能により、Red Hat OpenStack Image サービス (glance) のイメージデータを使用して、既存イメージを相互運用可能なイメージのインポートワークフローを使用してエッジにある複数の Ceph Storage ストアにコピーすることができます。

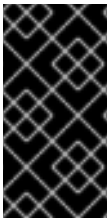
注記

イメージをエッジサイトのいずれかにコピーするためには、そのイメージは中央サイトに存在していなければなりません。既存のイメージを新たに追加したストアにコピーできるのは、イメージの所有者または管理者だけです。

--all-stores を **true** に設定するか、イメージデータを受け取る特定のストアを指定して、既存のイメージデータをコピーすることができます。

- **--all-stores** オプションのデフォルト設定は **false** です。**--all-stores** が **false** の場合は、**--stores STORE1,STORE2** を使用してイメージデータを受け取るストアを指定する必要があります。指定されたストアにイメージデータがすでに存在する場合、要求は失敗します。
- **--all-stores** を **true** に設定した場合、一部のストアにイメージデータがすでに存在していたら、それらのストアはリストから除外されます。

イメージデータを受け取るストアを指定すると、Image サービスは中央サイトからステージングエリアにデータをコピーします。続いて、Image サービスは相互運用可能なイメージのインポートワークフローを使用してイメージデータをインポートします。詳細は、複数ストアへのイメージのインポート を参照してください。



重要

Red Hat では、短時間に連続してイメージのコピー要求を行わないことを推奨します。同じイメージに対して短時間に 2 回イメージのコピー操作を行うと、競合状態が発生し予期せぬ結果を招きます。既存のイメージデータに影響はありませんが、新規ストアへのデータコピーに失敗します。

2.3.1. 全ストアへのイメージのコピー

利用可能なすべてのストアにイメージデータをコピーするには、以下の手順を使用します。

手順

1. 利用可能なすべてのストアにイメージデータをコピーします。

```
$ glance image-import IMAGE-ID \
--all-stores true \
--import-method copy-image
```

   IMAGE-ID をコピーするイメージの名前に置き換えます。

2. 利用可能なすべてのストアにイメージデータが正常に複製されたことを確認します。

```
$ glance image-list --include-stores
```

   イメージインポート操作のステータスを確認する方法については、イメージインポート操作の進捗の確認 を参照してください。

2.3.2. 特定ストアへのイメージのコピー

特定のストアにイメージデータをコピーするには、以下の手順を使用します。

手順

1. 特定のストアにイメージデータをコピーします。

```
$ glance image-import IMAGE-ID \
--stores STORE1,STORE2 \
--import-method copy-image
```

   - **IMAGE-ID** をコピーするイメージの名前に置き換えます。
   - **STORE1** および **STORE2** を、イメージデータをコピーするストアの名前に置き換えます。

2. 指定したストアにイメージデータが正常に複製されたことを確認します。

```
$ glance image-list --include-stores
```

   イメージインポート操作のステータスを確認する方法については、イメージインポート操作の進捗の確認 を参照してください。

2.4. 特定ストアからのイメージの削除

この機能により、OpenStack Image サービス (glance) を使用して、特定のストアにある既存のイメージコピーを削除することができます。

手順

特定のストアからイメージを削除します。

```
$ glance stores-delete --store _STORE_ID_ _IMAGE_ID_
```

- STORE_ID は、イメージのコピーを削除するストアの名前に置き換えます。
- **IMAGE_ID** を削除するイメージの ID に置き換えます。

警告

glance image-delete を使用すると、すべてのサイトでイメージが完全に削除されます。イメージのすべてのコピーに加えて、イメージインスタンスおよびメタデータが削除されます。

2.5. イメージの場所について

イメージは複数のサイトに存在できますが、それぞれのイメージは 1 つの UUID しか持ちません。イメージのメタデータには、各コピーの場所が含まれます。たとえば、2 つのエッジサイトに存在するイメージは、3 つの場所 (中央サイトおよび 2 つのエッジサイト) に単一の UUID として公開されます。

手順

1. イメージのコピーが存在するサイトを表示します。

```
$ glance image-show ID | grep "stores"

| stores |  default_backend,dcn1,dcn2
```

この例では、イメージは中央サイト (**default_backend**) ならびに 2 つのエッジサイト (**dcn1** および **dcn2**) に存在します。

2. あるいは、**--include-stores** オプションを指定して **glance image-list** コマンドを実行し、イメージが存在するサイトを表示することができます。

```
$ glance image-list --include-stores

| ID                                   | Name    | Stores

| 2bd882e7-1da0-4078-97fe-f1bb81f61b00 | cirros | default_backend,dcn1,dcn2
```

3. イメージの場所の属性をリスト表示し、それぞれの場所の詳細を表示します。

```
$ openstack image show ID -c properties

| properties |
```

```
(--- cut ---)
locations='[{'url': 'rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-
4078-97fe-f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}, {'url': 'rbd://63df2767-
8ddb-4e06-8186-8c155334f487/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap',
'metadata': {'store': 'dcn1'}}, {'url': 'rbd://1b324138-2ef9-4ef9-bd9e-
aa7e6d6ead78/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap', 'metadata': {'store':
'dcn2'}}]',
(--- cut --)
```

イメージの属性には、各イメージの場所ごとに異なる Ceph RBD URI が表示されます。

この例では、中央のイメージの場所の URI は以下のとおりです。

```
rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-4078-97fe-
f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}
```

URI は以下のデータで設定されます。

- **79b70c32-df46-4741-93c0-8118ae2ae284** は中央の Ceph FSID を表します。それぞれの Ceph クラスターは、固有の FSID を持ちます。
- すべてのサイトのデフォルト値は **images** です。これは、イメージが保存される Ceph プールを表します。
- **2bd882e7-1da0-4078-97fe-f1bb81f61b00** はイメージの UUID を表します。あるイメージの UUID は、場所に関係なく同一です。
- メタデータには、この場所がマッピングする glance ストアが表示されます。この例では、中央のハブサイトである **default_backend** にマッピングします。

付録A イメージの設定パラメーター

以下のキーは、**glance image-update** および **glance image-create** の両コマンドの **property** オプションに使用することができます。

```
$ glance image-update IMG-UUID --property architecture=x86_64
```

表A.1 属性のキー

対象コンポーネント	キー	説明	サポートされている値
すべて	**architecture**	ハイパーバイザーがサポートする必要のある CPU アーキテクチャー。たとえば、**x86_64**、**arm**、**ppc64** 等。マシンのアーキテクチャーを確認するには、**uname -m** を実行します。	• **alpha** - DEC 64 ビット RISC • **armv7l** - ARM Cortex-A7 MPCore • **cris** - Ethernet, Token Ring、AXis-Code Reduced Instruction Set • **i686** - Intel sixth-generation x86 (P6 マイクロアーキテクチャー) • **ia64** - Itanium • **lm32** - Lattice Micro32 • **m68k** - Motorola 68000 • **microblaze** - Xilinx 32 ビット FPGA (Big Endian) • **microblazeel** - Xilinx 32 ビット FPGA (Little Endian) • **mips** - MIPS 32 ビット RISC (Big Endian) • **mipsel** - MIPS 32 ビット RISC (Little Endian) • **mips64** - MIPS 64 ビット RISC (Big Endian) • **mips64el** - MIPS 64 ビット RISC (Little Endian) • **openrisc** - OpenCores RISC • **parisc** - HP Precision Architecture RISC • **parisc64** - HP Precision Architecture 64 ビット RISC • **ppc** - PowerPC 32 ビット • **ppc64** - PowerPC 64 ビット • **ppcemb** - PowerPC (Embedded 32 ビット)

対象コンポーネント	キー	説明	サポートされている値
			• **s390** - IBM Enterprise Systems Architecture/390 • **s390x** - S/390 64 ビット • **sh4** - SuperH SH-4 (Little Endian) • **sh4eb** - SuperH SH-4 (Big Endian) • **sparc** - Scalable Processor Architecture、32 ビット • **sparc64** - Scalable Processor Architecture、64 ビット • **unicore32** - Microprocessor Research and Development Center RISC Unicore32 • **x86_64** - IA-32 の 64 ビット拡張 • **xtensa** - Tensilica Xtensa 設定可能マイクロプロセッサーコア • **xtensaeb** - Tensilica Xtensa 設定可能マイクロプロセッサーコア (Big Endian)
すべて	**hypervisor_type**	ハイパーバイザーの種別	**kvm**、**vmware**
すべて	**instance_uuid**	スナップショットイメージの場合、このイメージを作成するのに使用したサーバーの UUID	有効なサーバーの UUID
すべて	**kernel_id**	AMI 形式のイメージをブートする際にカーネルとして使用する必要のある Image サービスに保管されているイメージの ID	有効なイメージ ID
すべて	**os_distro**	オペレーティングシステムのディストリビューションの小文字による一般名	• **arch** - Arch Linux。**archlinux** および **org.archlinux** は使用しないでください。 • **centos** - Community Enterprise Operating System。**org.centos** および **CentOS** は使用しないでください。 • **debian** - Debian。**Debian** および **org.debian** は使用しないでください。

対象コンポーネント	キー	説明	サポートされている値
			• **fedora**: Fedora。**org.fedora**、**org.fedoraproject** は使用しないでください。 • **freebsd**: FreeBSD。**org.freebsd**、**freeBSD**、**FreeBSD** は使用しないでください。 • **gentoo**: Gentoo Linux。**Gentoo** および **org.gentoo** は使用しないでください。 • **mandrake**: Mandrakelinux (MandrakeSoft) ディストリビューション。**mandrakelinux** および **MandrakeLinux** は使用しないでください。 • **mandriva**: Mandriva Linux。**mandrivalinux** は使用しないでください。 • **mes**: Mandriva Enterprise Server。**mandrivaent** および **mandrivaES** は使用しないでください。 • **msdos** Microsoft Disc Operating System。**ms-dos** は使用しないでください。 • **netbsd**: NetBSD。**NetBSD** および **org.netbsd** は使用しないでください。 • **netware**: Novell NetWare。**novell** および **NetWare** は使用しないでください。 • **openbsd**: OpenBSD。**OpenBSD** および **org.openbsd** は使用しないでください。 • **opensolaris**: OpenSolaris。**OpenSolaris** および **org.opensolaris** は使用しないでください。 • **opensuse**: openSUSE。**suse**、**SuSE**、**org.opensuse** は使用しないでください。 • **rhel**: Red Hat Enterprise Linux。**redhat**、**RedHat**、**com.redhat** は使用しないでください。 • **sled**: SUSE Linux Enterprise Desktop。**com.suse** は使用しないでください。 • **ubuntu**: Ubuntu。**Ubuntu**、**com.ubuntu**、**org.ubuntu**、**canonical** は使用しないでください。

対象コンポーネント	キー	説明	サポートされている値
			● **windows**: Microsoft ● **com.microsoft.server** は使用しないでください。
すべて	**os_version**	ディストリビューターによって指定されるオペレーティングシステムのバージョン	バージョン番号 (例:11.10)
すべて	**ramdisk_id**	AMI 形式のイメージをブートする際に ramdisk として使用する必要のある、Image サービスに保管されているイメージの ID	有効なイメージ ID
すべて	**vm_mode**	仮想マシンのモード。仮想マシンに使用されるホスト/ゲストの ABI (アプリケーションバイナリーインターフェイス) を示します。	**hvm**: 完全仮想化。これは QEMU および KVM で使用されるモードです。
libvirt API ドライバー	**hw_disk_bus**	ディスクデバイスの接続先となるディスクコントローラーのタイプを指定します。	**scsi**、**virtio**、**ide**、**usb** のいずれか。**iscsi** を使用している場合には、**hw_scsi_model** を **virtio-scsi** に設定する必要がある点に注意してください。
libvirt API ドライバー	**hw_cdrom_bus**	CD-ROM デバイスの接続先となるディスクコントローラーの種別を指定します。	**scsi**、**virtio**、**ide**、**usb** のいずれか。**iscsi** を指定する場合は、**hw_scsi_model** パラメーターを **virtio-scsi** に設定する必要があります。
libvirt API ドライバー	**hw_numa_nodes**	インスタンスに公開する NUMA ノードの数 (フレーバーの定義はオーバーライドしません)	整数

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバー	**hw_numa_cpus.0**	仮想 CPU N-M から NUMA ノード 0 へのマッピング (フレーバーの定義はオーバーライドしません)	整数のコンマ区切りリスト
libvirt API ドライバー	**hw_numa_cpus.1**	仮想 CPU N-M から NUMA ノード 1 へのマッピング (フレーバーの定義はオーバーライドしません)	整数のコンマ区切りリスト
libvirt API ドライバー	**hw_numa_mem.0**	N MB の RAM から NUMA ノード 0 へのマッピング (フレーバーの定義はオーバーライドしません)	整数
libvirt API ドライバー	**hw_numa_mem.1**	N MB の RAM から NUMA ノード 1 へのマッピング (フレーバーの定義はオーバーライドしません)	整数
libvirt API ドライバー	**hw_qemu_guest_agent**	ゲストエージェントのサポート。**yes** に設定し、かつ **qemu-ga** もインストールされている場合には、ファイルシステムが休止 (フリーズ) し、スナップショットが自動的に作成されます。	**yes** / **no**

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバー	**hw_rng_model**	乱数生成器をイメージのインスタンスに追加します。インスタンスのフレーバーを設定することにより、クラウド管理者は、デバイスの動作を有効化して制御することができます。デフォルトでは以下のように設定されます。 ● 乱数生成器は無効化されます。 ● /dev/random がデフォルトのエントロピーソースとして使用されます。物理ハードウェアの乱数生成器を指定するには、Compute 環境ファイルで **rng_dev_path** を/dev/hwrng に設定します。	**virtio** またはその他のサポートされているデバイス

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバー	**hw_scsi_model**	VirtIO SCSI (virtio-scsi) の使用を有効にして、コンピュートインスタンスのブロックデバイスアクセスを提供します。デフォルトでは、インスタンスは VirtIO Block (virtio-blk) を使用します。VirtIO SCSI は準仮想化 SCSI コントローラーデバイスで、より高いスケーラビリティーとパフォーマンスを提供し、高度な SCSI ハードウェアに対応します。	**virtio-scsi**

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバー	**hw_video_model**	仮想マシンインスタンスで使用するビデオデバイスドライバー	サポートされるドライバーのリストを優先される順に示します。 • **virtio**: (推奨) Gallium GPU 仕様の仮想 GPU で、OpenGL をレンダリングするのに VIRGL レンダラーを使用します。この GPU モデルはすべてのアーキテクチャーでサポートされ、ホストに専用の GPU がある場合には、ハードウェアアクセラレーションを利用することができます。詳細は、Virgil 3D GPU プロジェクト を参照してください。 • **qxl**: Spice または noVNC 環境用の高性能ドライバー • **cirrus**: レガシードライバー。**QXL** ドライバーが利用できない場合に使用します。 • **vga**: IBM Power 環境用にこのドライバーを使用します。 • **gop**: QEMU/KVM 環境ではサポートされません。 • **xen**: KVM 環境ではサポートされません。 • **vmvga**: レガシードライバー。このドライバーは使用しないでください。 • **none**: この値を使用して、仮想 GPU (vGPU) インスタンスのエミュレートされたグラフィックスまたはビデオを無効にします。この場合、ドライバーは別途設定されます。
libvirt API ドライバー	**hw_video_ram**	ビデオイメージの最大 RAM。フレーバーの **extra_specs** で **hw_video:ram_max_mb** の値が設定済みで、かつその値が **hw_video_ram** で設定されている値を上回る場合にのみ使用されます。	整数 (MB 単位。例: 64)

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバー	**hw_watchdog_action**	サーバーがハングした場合に指定したアクションを実行する仮想ハードウェアウォッチドッグデバイスを有効にします。このウォッチドッグは、i6300esb デバイスを使用します (PCI Intel 6300ESB をエミュレート)。**hw_watchdog_action** が指定されていない場合には、ウォッチドッグは無効になります。	• disabled: デバイスは接続されていません。イメージのフレーバーを使用して有効化されている場合でも、ユーザーがイメージのウォッチドッグを無効にすることができます。このパラメーターのデフォルト値は disabled です。 • reset: ゲストを強制的にリセットします。 • poweroff: ゲストの電源を強制的に切断します。 • pause: ゲストを一時停止します。 • none: ウォッチドッグを有効化するのみで、サーバーがハングした場合には何もしません。
libvirt API ドライバー	**os_command_line**	デフォルトではなく、libvirt ドライバーで使用されるカーネルコマンドライン。Linux Containers (LXC) の場合は、この値が初期化の引数として使用されます。このキーは、Amazon カーネル、ramdisk、またはマシンイメージ (aki、ari、または ami) にのみ有効です。	

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバーおよび VMware API ドライバー	**hw_vif_model**	使用する仮想ネットワークインターフェイスデバイスのモデルを指定します。	設定したハイパーバイザーによって有効なオプションは異なります。 ● KVM および QEMU: e1000、ne2k_pci、pcnet、rtl8139、virtio ● VMware: e1000、e1000e、VirtualE1000、VirtualE1000e、VirtualPCNet32、VirtualSriovEthernetCard、VirtualVmxnet ● Xen: e1000、netfront、ne2k_pci、pcnet、rtl8139
VMware API ドライバー	**vmware_adaptertype**	ハイパーバイザーが使用する仮想 SCSI または IDE コントローラー	**lsiLogic**、**busLogic**、または **ide**
VMware API ドライバー	**vmware_ostype**	イメージにインストールされているオペレーティングシステムを示す VMware GuestID。この値は、仮想マシンの作成時にハイパーバイザーに渡されます。指定しなかった場合には、このキーの値はデフォルトの **otherGuest** に設定されます。	詳細は、Images with VMware vSphere を参照してください。
VMware API ドライバー	**vmware_image_version**	現在は使用されていません。	**1**

対象コンポーネント	キー	説明	サポートされている値
XenAPI ドライバー	**auto_disk_config**	true に指定した場合には、ディスク上のルートパーティションは、インスタンスがブートする前に自動的にリサイズされます。この値は、Xen ベースのハイパーバイザーを XenAPI ドライバーと共に使用する場合にのみ Compute サービスによって考慮されます。Compute サービスは、イメージに単一のパーティションがあり、かつそのパーティションが **ext3** または **ext4** のフォーマットの場合にのみリサイズを試みます。	**true** / **false**

対象コンポーネント	キー	説明	サポートされている値
libvirt API ドライバーおよび XenAPI ドライバー	**os_type**	イメージ上にインストールされるオペレーティングシステム。XenAPI ドライバーには、イメージの **os_type** パラメーターの値によって異なるアクションを実行するロジックが組み込まれています。たとえば、**os_type=windows** イメージの場合には、Linux スワップパーティションの代わりに、FAT32 ベースのスワップパーティションを作成し、挿入されるホスト名を 16 文字未満に制限します。	**linux** または **windows**